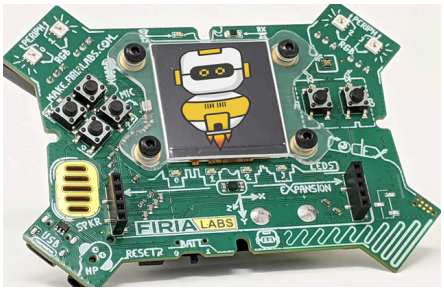
Python with CodeX

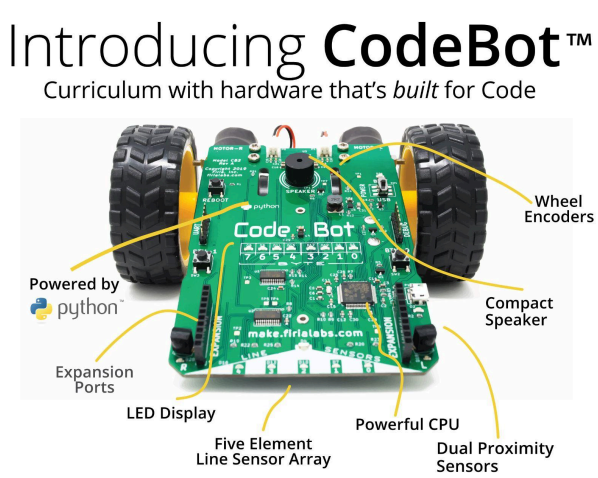


Students learn best when they are engaged in project-based hands-on physical computing. With the CodeX device and our CodeSpace learning environment, watch students step up to computer science with real-world, text-based Python coding! Use our authentic and motivating curriculum to ignite an enduring passion for creating, computational thinking, and a genuine love for coding and robotics.

Python with CodeX is a gentle introduction to Python programming with a physical computing device. This curriculum is a drop-in unit that introduces basic programming concepts while mirroring industry standards, for any entry-level CS or CTE course.

CodeX Missions in CodeSpace		South Dakota CTE Standards
Mission 1	Welcome -- learn about CodeSpace	**Course alignment:** Computer Science Essentials, Computer Science Principles
Mission 2	Introducing CodeX -- learn about the CodeX	Standards: CSP 1.1, 1.2, 1.3
Mission 3	Light Show -- turn pixel LEDs various colors	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 4	Display Games -- lighting pixel game	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 5	Micro Musician -- use sound clips	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Remix #1	Create original code from missions 3-5	
Mission 6	Heartbeat -- introduces a while loop	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 7	Personal Billboard -- introduces a list	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 8	Answer Bot -- uses a list	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Remix #2	Create original code from missions 6-8	
Mission 9	Game Spinner -- introduces functions with parameters	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 10	Reaction Time -- use the computer's clock	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 11	Spirit Level -- use the accelerometer to turn the CodeX into a digital level	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
Mission 12	Night Light -- use the light sensor to turn on and off the LED pixels	Standards: CSP 4.1, 4.2, 4.3, 4.4, 4.5, 4.6

Python with Robots



Students thrive in an environment that fosters project-based, hands-on physical computing, and the Python with Robots curriculum, paired with the innovative CodeBot, is designed to deliver just that. With the CodeBot device and our CodeSpace learning environment, witness students dive into the world of computer science through real-world, text-based Python coding! Our curriculum is crafted to be authentic and motivating, aiming to spark a lasting passion for creation, computational thinking, and a genuine love for both coding and robotics. Experience the future of STEM education with CodeBot—where learning is interactive, meaningful, and geared towards shaping the tech leaders of tomorrow.

Python with Robots is an intermediate level Python programming course using a physical computing device with wheels. This curriculum is a drop-in unit that covers basic through intermediate programming concepts for most beginner or intermediate CS or CTE courses. This course is a step-up in rigor from our Python with CodeX course.

Python with Robots Missions in CodeSpace		South Dakota CTE Standards
Project 1	First Steps - hands on tour of CodeBot	**Course alignment:** Computer Programming I, Advanced Computer Programming
Project 2	Time & Motion - motor control, LED lights, button switches; if statements	Standards: CP 1.1, 1.2, 2.1, 2.2, 2.3, 4.3
Project 3	Animatronics - use speaker to add sounds	Standards: CP 3.1, 3.2, 3.3
Project 4	Fence Patrol - Line Sensors for boundary detection	Standards: CP 3.4, 4.1, 4.2
Project 5	Line Follower - Line sensors & lists	Standards: CP 3.3, 3.4, 4.2
Remix #1	Create original code from Projects 2-5	
Project 6	Hot Pursuit - Proximity sensors and	Standards: CP 3.3, 3.4, 4.2

	pursue a moving target	
Project 7	Navigation - Wheel encoders & Functions	CP 3.3, 3.4, 4.2, 4.3
Project 8	All Systems Go - monitor battery voltage, system temperature, & physical orientation with Accelerometer	CP 3.3, 3.4, 4.2, 4.3
Remix #2	Create original code from Projects 6-8	
Project 9	Obstacle Course - “object avoidance challenge” - combine all skills to tackle an obstacle course	CP 1, 2, 3, 4
Project 10	Multitasking - Botservices, event-driven programming; combine all skills to perform multiple tasks at the same time	CP 1, 2, 3, 4

Level-1 Python with Virtual Robotics

Step into the future of STEM education with Level-1 Python with Virtual Robotics—a transformative experience that integrates the CodeBot Sim device with our CodeSpace learning environment. In this dynamic curriculum, students engage in a project-based, simulated approach to physical computing, exploring computer science through real-world Python coding. From mastering Python basics to tackling real-world challenges, students delve into key computer science concepts like loops, function definitions, API library usage, and conditional logic. The distinctive focus on virtual physical computing ensures a solid grasp of robotics principles applicable in the physical realm. Upon completion, students are well-prepared for the PCEP: Certified Entry-Level Python Programmer certification and Certiport IT Specialist/Python exams, empowering them as the tech leaders of tomorrow. Experience interactive and meaningful learning with the Sim CodeBot, shaping the future of STEM education.

Firia Labs Python Level-1 Learning Objectives

Level-1 Python with Virtual Robotics is an advanced level Python programming course using a virtual simulated computing device. This curriculum is a drop-in unit that covers advanced programming concepts aligning with advanced computer programming or CTE courses. This is a capstone course in our Python learning pathway, leading to a Python certification.

*This is a unified set of Learning Objectives covering the requirements of both Certiport and OpenEDG.

Ref	Category	Concept	Focus Mission	Other Missions
1.1	builtins	input()	Line Sensors	Scoreboard
1.2	builtins	len()	Robot Metronome	Scoreboard
1.3	builtins	built-in functions	Dance Bot	
1.4	builtins	print() with sep, end params	Dance Bot	

2.1	concepts	Interpreter vs Compiler	Teacher Manual	
2.2	concepts	Source code vs Object (machine) code	Teacher Manual	
2.3	concepts	coding style, PEP8 basics	Teacher Manual	
2.4	concepts	Errors: Syntax, Runtime, Logic	Teacher Manual	Scoreboard
3.1	core	None	(future)	
3.2	core	identity operator: 'is'	Eternal Flame	
3.3	core	using del to "undefined" variables	(future)	
3.4	core	type inspection using type() function	Eternal Flame	
3.5	core	pass	Cyber Storm	
3.6	core	Using help() on the REPL	(future)	
3.7	core	Backslash line continuation	(future)	
3.8	core	multiple assignment (unpacking)	Music Box	
3.9	core	conditional statements: elif, else	Fido Fetch	
3.10	core	augmented assignments	Go the Distance	
3.11	core	type conversion: int()	Music Box	Rock Climber and Combo Lock
3.12	core	global vs local scope, global keyword	Line Following	
3.13	core	bool	Robot Metronome	
3.14	core	conditional statements: if	Robot Metronome	
3.15	core	Keywords vs user-defined variable names	Dance Bot	
3.16	core	Indentation	Dance Bot	
3.17	core	comments	Light the Way	
4.1	exceptions	exception handling: try, except	Scoreboard	
4.2	exceptions	exception handling: else, finally	Scoreboard	
4.3	exceptions	raising exceptions: raise	Scoreboard	
5.1	files	File I/O: append, with	Cyber Storm	

5.2	files	File existence check, deletion	Cyber Storm	
5.3	files	File I/O: open, close, read, write	Music Box	
6.1	functions	recursion	(future)	
6.2	functions	parameters vs arguments	Boundary Patrol	
6.3	functions	positional vs keyword arguments	Boundary Patrol	
6.4	functions	function return values	Line Sensors	
6.5	functions	default function parameters	Dance Bot	Boundary Patrol
6.6	functions	defining functions	Dance Bot	
7.1	loops	continue	(future)	
7.2	loops	while-else, for-else	(future)	
7.3	loops	using for loop to iterate over string	(future)	
7.4	loops	multiple assignment in for loop	Music Box	
7.5	loops	using for loop to iterate over list	Music Box	
7.6	loops	break	Line Sensors	Cyber Storm
7.7	loops	while loop	Dance Bot	
7.8	loops	for loop, range()	Dance Bot	
8.1	math	float (type and coercion/ctor)	Eternal Flame	
8.2	math	Scientific notation	(future)	
8.3	math	bitwise operators: ~	(future)	
8.4	math	bitwise operators: &	Combination Lock	
8.5	math	bitwise operators: |	Combination Lock	Scoreboard
8.6	math	bitwise operators: ^	Combination Lock	
8.7	math	int (type and coercion/ctor)	Music Box	
8.8	math	Modulo %	Runway Ops	
8.9	math	Numeric multiply * operator	Go the Distance	
8.10	math	Numeric divide / operator	Go the Distance	
8.11	math	Integer division //	Go the Distance	Runway Ops

8.12	math	hex and octal literals	Combination Lock	
8.13	math	Power ** operator	Combination Lock	Runway Ops
8.14	math	boolean 'and'	Line Following	
8.15	math	boolean 'or'	Line Following	
8.16	math	operator precedence	Robot Metronome	
8.17	math	bitwise shifts: << >>	Robot Metronome	Combination Lock & Scoreboard
8.18	math	boolean 'not'	Robot Metronome	Scoreboard
8.19	math	comparison operators	Robot Metronome	
8.20	math	binary literals	Light the Way	Combination Lock
9.1	modules	datetime module (strftime, strptime)	Time Machine	
9.2	modules	math module	Rock Climber	
9.3	modules	random module	Eternal Flame	
9.4	modules	import of modules	Light the Way	
9.5	modules	Using unittest	Teacher Manual	
9.6	modules	os, sys, os.path, io	Teacher Manual	Cyber Storm
10.1	sequences	using list() constructor	Traffic Light	
10.2	sequences	using tuple() constructor	Traffic Light	
10.3	sequences	containment tests: 'in' and 'not in'	Cyber Storm	
10.4	sequences	dictionary: copy() method	Traffic Light	
10.5	sequences	list: copy() method and [:] to copy	Traffic Light	
10.6	sequences	slicing lists	Traffic Light	
10.7	sequences	copying a list	Traffic Light	
10.8	sequences	negative indices	Eternal Flame	
10.9	sequences	list: extend()	Traffic Light	
10.10	sequences	list operator: +	Traffic Light	
10.11	sequences	list operator: *	Runway Ops	

10.12	sequences	list: insert()	Traffic Light	
10.13	sequences	list: remove()	Traffic Light	
10.14	sequences	list: del (index or slice)	Traffic Light	
10.15	sequences	list/tuple: index()	Traffic Light	
10.16	sequences	list/tuple: sorted(), reversed()	Eternal Flame	
10.17	sequences	dictionary: keys(), items(), values()	(future)	
10.18	sequences	list: sort()	Eternal Flame	
10.19	sequences	list: append()	Music Box	
10.20	sequences	using dict() constructor	(future)	
10.21	sequences	tuple: literals and usage	Line Following	
10.22	sequences	dictionary: literals and usage	Line Following	
10.23	sequences	list comprehensions	Line Following	
10.24	sequences	Nested lists/tuples: matrices	Line Sensors	
10.25	sequences	list: literals and usage	Robot Metronome	
11.1	strings	string (type and coercion/ctor)	Cyber Storm	
11.2	strings	slicing strings	Cyber Storm	
11.3	strings	string escape sequences	Cyber Storm	
11.4	strings	multiline strings	Music Box	
11.5	strings	string formatting with f-strings	Go the Distance	
11.6	strings	string operator: +	Scoreboard	Cyber Storm
11.7	strings	string operator: *	Rock Climber	
11.8	strings	type conversion: str()	Time Machine	Combination Lock
11.9	strings	string formatting with string.format()	Rock Climber	
12.1	tools	docstrings	Boundary Patrol	
12.2	tools	Using pydoc	Teacher Manual	